Secure Reconfigurable Computing

David W. Jensen, Ph.D.
David A. Greve
Matthew M. Wilding, Ph.D.

Rockwell Collins Advanced Technology Center
Cedar Rapids, Iowa
{dwjensen,dagreve,mmwildin}@collins.rockwell.com

Abstract

Three technologies must be advanced to enable the migration of reconfigurable computing from research to security and safety critical applications. Those technologies are rapid dynamic reconfiguration, multiple user support, and secure application separation. All three technologies are necessary to meet the requirements of future avionics, security, and defense applications.

In this paper, we present a reconfigurable computing architecture that is explicitly secure for multiple user environments and supports varying degrees of criticality and privilege. At the core of our secure reconfigurable architecture is a real-time Multiple Virtual Machine (MVM) model in a direct execution JVM microprocessor [AW97,DAG98A]. Our architecture provides hardware-enforced guarantees of resource separation. We have extended this separation guarantee to support reconfigurable logic devices.

Implementations of our architecture can be verified to be safe and secure [JMR98]. We outline our formal verification techniques, which are published and can be applied to modern safety-critical and security-critical development environments [DAG98B, SPM96]. We detail an approach for formally validating that our architecture enforces separation.

An architecture should be developed with open system standards to exploit future technological advances. For that reason, we employ Java™ as a cornerstone of our design. In our vision, the same Java™ classfiles can be used on a computer system with or without reconfigurable computing capabilities. We use the Java™ software method invocation interface to execute hardware algorithms on the reconfigurable computing elements. We present these concepts and preliminary results from our system simulations [SAS98].

1. Introduction

The Collins division of Rockwell has been designing and producing avionics quality microprocessors for 25 years. A heritage of Collins microprocessors exists in scores of avionics platforms. One product, the FCP 2000, is the only microprocessor certified by the FAA for multiple similar processor applications because of its level of verification. A more recent product is the JEM1 microprocessor, which is the first hardware implementation of the Java Virtual Machine (JVM) instruction set [AW97].

The Adaptive Computing System (ACS) program has provided significant guidance and created advancements for reconfigurable computing architectures [ACS98]. We describe in this paper an architecture that builds on these efforts [LAJ98] and explicitly supports three key ACS technologies: rapid dynamic reconfiguration, multiple application support, and secure application separation. Successfully providing these three technologies will enable the migration of reconfigurable computing from research to military and commercial development.

™ - Java™ is trademarked by Sun Microsystems, Incorporated.

Our reconfigurable computing system design is unique in its explicit support of multiple applications and its secure separation of those applications. This system uses a direct-execution Java™ microprocessor, the JEM™. Java™ provides an open system standard for our software development *and* our hardware configurations. We intend to apply recent Rockwell Collins advances in verification to ensure critical system properties.

2. Secure Reconfigurable Architecture

We have created a reconfigurable computer architecture that is explicitly secure. The premise of this architecture is conceptually simple. Modern general-purpose microprocessors often include a memory management unit. This unit monitors all memory accesses for all processes executing on the system. The unit only allows each process access to specific ranges (or partitions) of memory. This partitioning ensures that processes cannot affect the memory space of other processes. If an architecture memory-maps all reconfigurable logic (both the configuration and access logic), then those Reconfigurable Computing Elements (RCEs) become explicitly partitioned. We illustrate this architectural concept in Figure 1.

Our research architecture presents an innovative approach to providing runtime support for embedded systems based on widely-adopted industry standards (e.g., Java™) coupled with our proven implementation expertise for efficient, low-power, small footprint, ruggedized avionics systems; see Figure 2. Rockwell Collins has developed the world's first direct execution Java™ microprocessor, the JEM1. The JEM1 features efficient support of object-oriented constructs; very low thread latency; a runtime system written completely in Java™; and a uniquely capable static linker which pre-resolves Java™ references, eliminates unnecessary methods, fields, constants, and classes, and produces ROMable images. The JEM™ platform is unique in its efficient support for the execution of Java™ bytecodes, its emphasis on hard real time, and its safe and secure multiple virtual machine (MVM) execution.

We intend to add reconfigurable computing to our JEM2 system architecture; see Figure 1. Our reconfigurable computing system places the host processor and a set of RCEs on a local bus. A Partition Management Unit (PMU) separates the local bus from the memory subsystem. The PMU monitors all memory transactions, whether originating from the host microprocessor or from one of the RCEs, to ensure that the requesting task has the authority to perform the transaction. If not, the PMU inhibits the transaction. Only the run time executive has authority to configure the RCEs and the RCEs can only modify devices and memory in their partitions. These simple, PMU-enforced transactions ensure that no processes or RCE algorithms will affect other processes or the runtime executive. Our Partition Management Unit (PMU) is the cornerstone of our secure reconfigurable architecture.

An impact of this design decision is that all RCEs must be configured to use the local bus to access and update memory values. This restriction limits the flexibility of the RCEs, but offers a stable interface for the development

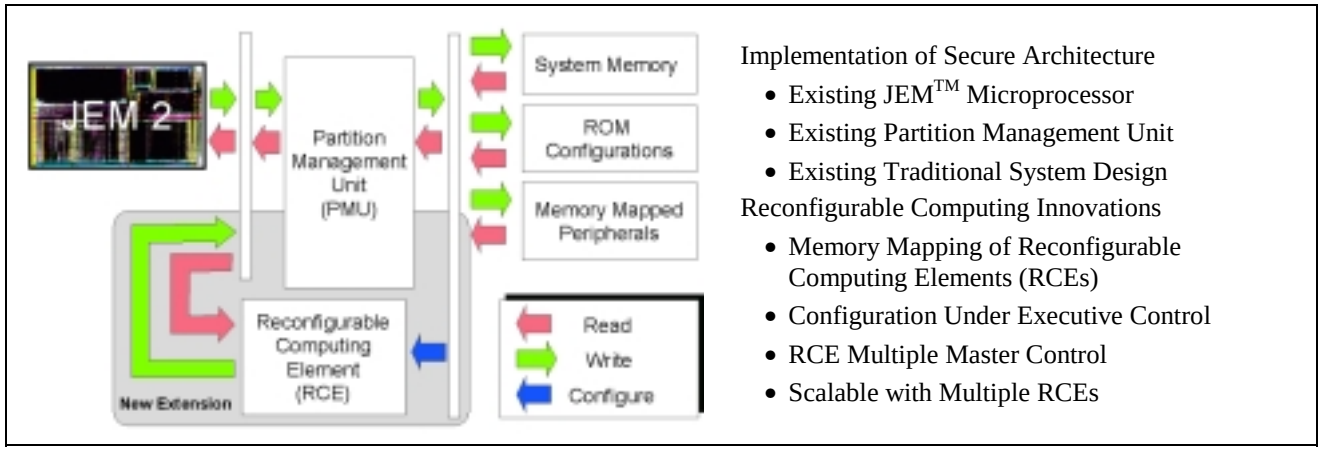


Figure 1. Architecture Extensions Required for Secure Reconfigurable Computing

™ - JEM™ is trademarked by Rockwell Collins, Incorporated

JEM™ Microprocessor Family Features	Characteristic
Efficient, low power, small footprint, ruggedized system	Versatility
Widely adopted industry standards	Compatibility
Networked controller	Network capability
Reconfigurable, dynamic loading	Reconfigurability
Safe and secure partitioning	Security
Software component substrate	Extensibility

Figure 2. Rockwell Collins JEM™ Microprocessor Family Features

of future reconfigurable computing elements. This stable interface also enables the development of hardware algorithms to a standard open-system interface. For local bus control, we intend to construct partition-aware arbitration logic that allows only certain entities to arbitrate for control. This control is dependent on the active partition. By disallowing access to the local bus, this scheme suspends the operation of certain components in favor of others in the same way that software processes are suspended during a processor context switch. Initially, we do not intend to share RCEs between partitions and state saving and restoration will not be required.

3. Multiple Application Support

Our current development uses the second generation JEM2 microprocessor, which is designed specifically to support a partition management unit (PMU). The PMU and the JEM2 runtime environment provide secure partitioning (or separation) of Java™ resources. We have implemented a Multiple Virtual Machine (MVM) model in the JEM2 microprocessor, which will form the core of the secure reconfigurable architecture. This model guarantees that multiple programs can execute concurrently, with hardware-enforced guarantees of resource isolation.

The PMU ensures separation of applications; see Figure 3. The separation required by the Federal Aviation Administration (FAA) for safety critical applications is similar to the separation required by National Security Agency (NSA) for multiple level security [JMR98]. Applications that require separation from one another will be relegated to separate partitions. The PMU will be programmed to provide each partition access only to it allocated resources. Resources include memory space, processing time, and RCEs.

The PMU to be used in this architecture is similar to the Memory Management Units (MMUs) found on many traditional computer systems. However, while typical MMUs provide protection at the task level, the PMU is designed to provide protection at the partition level, where each partition may be thought of as a complete Java™ Virtual Machine (JVM).

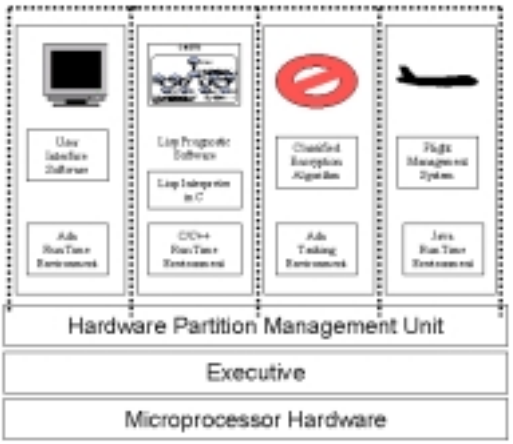
Another distinction between the PMU and a typical MMU is that the PMU is responsible for enforcing temporal isolation of the various partitions. The PMU guarantees that each partition consumes exactly its allocated share of the processing time. It monitors the processor via a watchdog timer and generates a non-maskable partition interrupt to force synchronization. This temporal partitioning allows the system designer to enforce not only worst case timing but also best case timing. This “invariant performance” is at the heart of our contract with the application developer [MMW99]. It allows us to guarantee that the operation of an application in this partition will be absolutely independent of the other partitions. Thus, any validation or verification work performed on an application in isolation will be guaranteed to hold in a composed system. It is the extension of this work to the reconfigurable computing elements that allows us to make strong claims regarding the safety and security of such a system.

The RCEs are resources that can be allocated to specific applications (partitions) and treated as logical extensions of the software executing on the JEM1. As such, the RCE will be allowed to make memory transactions only during those times allocated to its associated software partition. Because the PMU monitors all transactions from the RCE, it will allow the RCE access to only the memory space allocated to its owning partition. When the partition and its RCE is idle, the RCE will be denied access to the local bus and will be incapable of illicit monitoring of other partitions. In this way, the RCE will be unable to impact the activity of any other application in any other partition. Given this design, the system designer can ensure that malicious applications and RCEs cannot corrupt or monitor the operation of another application or RCE. We illustrate many of these concepts in Figure 3.

4. Secure Partitioning

Although it is theoretically possible to approach the high levels of assurance desired for this program using commercial microprocessors, the JEM™ microprocessor has been specifically designed to incorporate features supporting safety and security while providing rapid partition context switching. By incorporating features that support strict time and space partitioning, we can build systems that, by construction, provide a high degree of safety and security.

Our target applications for multiple-user reconfigurable computing platforms require the highest level of assurance. Our system is designed to control partitioning despite the extreme volatility of reconfigurable systems, but we recognize that subtle problems in our implementation could potentially have catastrophic effects. Because verification costs dominate the total development cost of safety-critical software, Rockwell Collins has invested heavily in modular verification. We expect to provide a contract with developers that will allow them to verify their software once in isolation and apply the results of that verification to systems composed of multiple applications of varying levels of criticality. Using this common model approach validates the model through its use as a simulator and allows us to formally verify properties of the system.



- Rockwell Collins has developed a Partition Management Unit as an integral part of the JEM™ microprocessor.
- This hardware memory and partition management unit ensures separation of applications.
- The separation required by the FAA for safety critical applications is analogous to the separation required by NSA for multiple level security.

Figure 3. Multiple Application Support

Our reconfigurable architecture supports multiple applications with varying degrees of criticality and varying degrees of privilege. It maintains multiple application separation whereby each application retains all its verified properties. It supports these applications in a readily composable form such that some applications can be added or removed without impacting the verified properties of the remaining applications.

We plan to analyze our implementation to ensure that its design enforces the separation required to support multiple users and multiple security levels in an adaptive computing system. Any approach to accomplishing this must address each of three fundamental challenges: model construction and validation, separation properties, and verification of model properties. We describe our technical approach to each of these challenges.

5. Formal Verification

The goal of verification is to analyze a system model and demonstrate that a complex design has certain required properties. Such verification has been accomplished on a variety of different targets and is similar to our verification approach [WRB89A, WRB89B, RSB96, SPM96, MMW97]. In particular, the use of so-called "mechanical theorem provers" as digital design tools is an active area of research, particularly with regard to how these tools can be integrated into the fast-changing world of digital design [DSH98]. Our current verification efforts are distinguished from most similar work in two main respects: our modeling approach allows for both analysis and model validation, and our use of automated analysis. Projects that have used reasoning tools to establish the correctness of computer system designs – both at Rockwell Collins and elsewhere – have often been very labor-intensive, and the analysis sensitive to minor changes in the system's specification or implementation [MMW97, DAG98B]. We believe that our emphasis on analysis automation will help overcome these challenges.

5.1 Model Construction and Validation

We wish to guarantee that our reconfigurable system safely and securely supports multiple users at multiple security levels. The first hurdle is to identify what description of the implementation we use to perform the analysis. This is a deceptively difficult challenge, because the model we create must not only support our analysis but must also have sufficient fidelity that we are confident that our analysis applies to the actual system.

There are many languages and tools for modeling and analysis, and we have used some of them in our work. In a Collins microprocessor verification effort [SPM99], we used the PVS theorem proving system's logic [SO98] to describe the operation of the microprocessor's microarchitecture. We used its theorem prover to characterize and verify the operation of much of the microprocessor's microcode. We believe, however, that this possible approach is not ideal for our system because the model must be validated and be part of the development process. Another approach for providing a system model is to use the system design directly. A VHDL description of the processor element, for example, obviously provides a reliable description of the operation of that processor. The kinds of errors that could cause discrepancies between the model and the actual system, such as fabrication errors, are already considered in the development process. This kind of model is inherently self-validating, and because the design is an artifact of the development process it will be maintained as a matter of course.

We intend to build a simulator that can serve as an analyzable artifact to verify the system; see Figure 4. We will use the simulator to develop the system implementation, and it will be validated by its use by the system developers who are most familiar with its expected operation. The simulator will then serve as a model for our analysis. As an artifact of the design process, the model will not become out-of-date or otherwise fall out of sync with the actual design, and its use as a simulator during development will provide a high level of validation. Further, it will be at a higher level of abstraction than the design, because it will describe the expected behavior of the system rather than its implementation in hardware, which will simplify analysis.

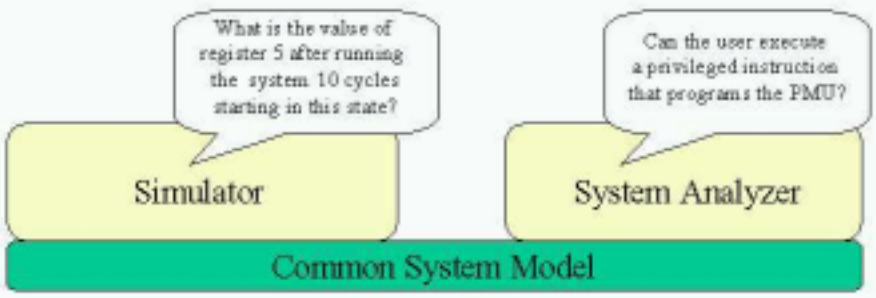
A significant challenge is to provide adequate simulator performance to integrate it into a real design environment. We have investigated combining simulator and analysis models in the context of industrial microprocessor development and have shown that this can provide simulator performance close to conventional simulators written in the C programming language [MMW98A].

5.2 Separation Properties

Our system will use a Partition Management Unit (PMU) to separate applications with multiple security or safety levels; see §3. We will determine a policy under which users are to be separated, and how that policy translates into specific system implementation properties that can be checked against our implementation. In this section we discuss safety-related work that is relevant to this problem and describe our approach to this challenge.

We have faced the need for separation in the context of the construction and certification of integrated modular avionics (IMA). Broadly speaking, we have taken a very static, deterministic approach that we call *invariant performance* [MMW99]. Our approach is also reported in [JMR98], where it is referred to as the “gold standard” partitioning property. The idea is to guarantee each partition predefined memory and CPU resources so that it can be integrated with other applications with no effect on its behavior. This approach allows separate verification of applications, because an application can be verified within a partition in the absence of other applications with confidence that the same performance will be seen after integration. Our IMA approach to guaranteeing safety properties is also useful for demonstrating security properties, because it requires a completely static architecture. Consequently, we expect the technique we have developed to verify separation in a safety context to translate well to security needs.

It will be possible to analyze the security requirements of our architecture and derive a set of implementation properties that are necessary to achieve these requirements. For example, our reconfigurable system will have multiple users, and the PMU ensures that the memory spaces of the users are distinct in order to protect the memory-mapped FPGAs of other users. However, the PMU can be programmed by the processor when the processor is in a privileged mode, and so perhaps a user could connive to eliminate the check. We may validate this property by analyzing and guaranteeing that in our reconfigurable system the processor cannot execute a privileged instruction when executing a user's code. We will build our reconfigurable system with static resource allocations and will derive a list of crucial properties that must hold of an implementation.



- Efficient execution (near compiled C speed)
- Tool for system developer
- Easily validate new extensions
- Push-button verification of requirements
- Validate system wide characteristics
- Common model used with simulator

Figure 4. The Use of a Single Model for Simulation and Analysis Provides Model Validation

5.3 Verification of Properties

We have described in previous sections our approach to modeling our reconfigurable system and how we will validate it, and generally some of its required properties. In this section, we describe our approach to ensuring that these properties hold of the model.

Of course, computer system correctness is notoriously difficult to establish. Analysis of computer system models has the potential to improve the reliability of computer system designs, but such analysis is often very complex and hard to get right. Mechanical tools such as PVS and ACL2 can help overcome both of these problems by automating the analysis, but their use can be time-consuming. For realistic-sized applications, the inevitable mistakes in proof development or changes to system design or specification will be very disruptive.

We have pursued several approaches to make the analysis of real computer systems more practical. Our basic approach is to make the analysis as automatic as possible. This allows for the establishment of correctness properties initially, but even more importantly it makes realistic the guarantee of correctness properties in the face of system and specification changes. Our development of "robust" software execution analysis is described in [MMW97]. By adding certain sound reasoning rules we make the analysis of code fragments nearly automatic. We developed another tool [DAG98B] that automatically derives a symbolic expression for each sequence of the microcode for the JEM1 [AW97,DAG98A].

Using these approaches to automation, we intend to develop "push-button" analysis of some security properties of the system model. These proofs will regenerate with little or no manual effort when the model changes in an irrelevant way, and will rely on symbolic simulation of the processor and PMU components. For example, the property described in the previous section – that a user can not execute a privileged instruction – will be demonstrated by showing that no processor instruction that the user can execute can operate in a privileged way.

Our approach to verifying correct reconfigurable and secure operation will support the development of a dependable multiple user reconfigurable system. We will build an analyzable model of our reconfigurable system that is validated through its use as an efficient simulator by system developers [MMW98A]. Our current work to build dependable integrated modular avionics through completely static resource scheduling [MMW99] will help guide our derivation of needed system properties. Our recent advances in the microprocessor verification through symbolic simulation are required to accomplish the "push-button" analysis [DAG98B]. Our verification and validation approach is experimental, but it is achievable in part because it leverages recent Rockwell Collins research.

6. Open System

The United States Government has defined its requirement of open systems as the ability to provide portability of software across standard system platforms, interoperability between applications, connectivity between systems, flexibility in the management of the information systems' resource and much greater choice in systems procurement [OGO98]. Compatibility, portability and interoperability are central. Open systems allow users to communicate and share data across different platforms. Analyzing these definitions, we contend that the key benefits of having an "open" reconfigurable system will be portability, compatibility, and supportability.

We intend to provide these four open-system benefits on our secure reconfigurable computing system. We recognize that a system must be developed with open system interfaces, software, and hardware configurations to exploit future technology advances. By using open system standards it is possible to leverage a wide range of work and always provide the best system solution.

Portability allows the same software to run on multiple platforms. We intend to employ Java™ as our primary software application environment. Applications written for the system and compiled into class files will be portable between various computer systems. The behavior of the application will be independent of whether or not the computer system provides reconfigurable computing capabilities. We will use the same Java™ method interface to execute hardware algorithms on the reconfigurable computing elements. Using this open interface will enable implementation independence from the application developer's perspective. Our secure reconfigurable computing system will execute code compiled to JVM bytecodes independent of the development platform.

JVM classfiles from any system will run on our reconfigurable system and the classfiles from our system will run on any other JVM supported system; see Figure 5. Many developers are familiar with Java™ and its JVM because is an open specification. Java™ is becoming a universal medium for exchange - not only for software, but for binaries, architecture, ideas, and hardware definition. The JEM™ processor was designed to execute the Java™ virtual machine instruction set natively. This provides for high performance on object oriented code. The use of the JVM as the development platform provides other advantages. It provides a widely available host based development environment for code developed for execution on our system. Host based development has many advantages including extensive debugging and verification capabilities and minimal reliance on the availability of target systems.

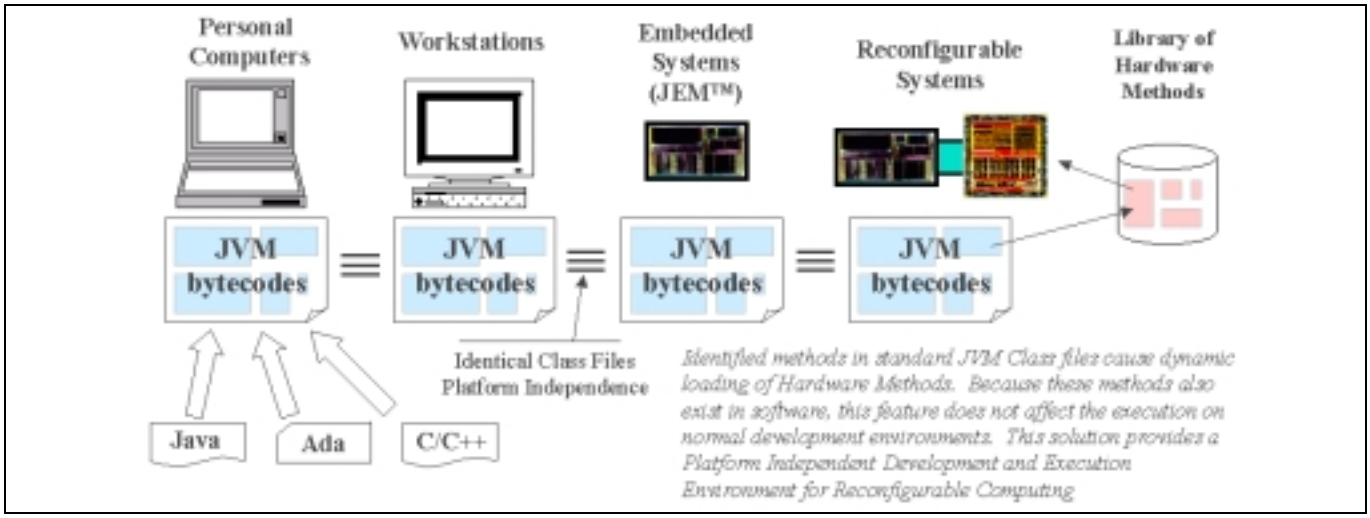


Figure 5. Portability of Java™ Bytecodes Across all Platforms

We will use the static linker of the JEM™ to recognize and resolve the hardware methods and make links to these routines. The static linker performs functions similar to that of the dynamic loader component of a software Java™ virtual machine interpreter, but does them prior to Java™ execution. A constraint on this portability is that the reconfigurable architecture will only be utilized if the program uses the methods defined in the library of hardware methods. These routines will be built from compiled VHDL code or use software translation tools to maintain portability to multiple FPGA hardware. Exploiting the object-oriented nature of Java™ allows the system to choose an RCE most applicable to the data upon which it must operate. The JVM method invocation interface provides a clean contract between the software developer and the hardware implementation. We intend to exploit the JVM calling convention to provide a nearly seamless integration between the hardware implementation and the application software.

Compatibility, in our reconfigurable computing environment, is attained with the FPGA method invocation interface and its compatibility with the traditional software JVM method invocation interface. The invocation of FPGA methods will be transparent to users and portable across many systems. We intend to pass parameters to hardware methods through the same JVM method invocation protocol. Parameters are pushed onto the JVM process stack and results are returned on the stack. We have designed and simulated an RCE using this interface [SAS98].

A system architecture should be independent of the architecture of the reprogrammable devices in order to leverage advances in technologies. To remain compatible with all other JVM platforms, invocation of FPGA methods will simply use the JVM method innovation interface. This can be accomplished by using the JEM™ static linker to create linkage to those routines. The method invocation interface is used between the JEM™ and the hardware methods in the RCEs or FPGAs. Because the interface is completely device independent, it allows for the use of a broad set of components. For this reason, our reconfigurable system can be implemented using a wide variety of RCEs and therefore is capable of leveraging any technological improvements in the intrinsic performance capabilities of the RCEs themselves, such as reduced programming time and modular programming capability.

Obviously, the library of hardware methods will be FPGA dependent. However, hardware methods can be compiled from FPGA-independent VHDL to the FPGA dependent programming information. Translating programming languages directly to VHDL is a recent technology [RG98] that will aid the development of these hardware methods. Our research team has developed a manual process to convert JVM bytecodes to VHDL [SAS98]. We have applied this process to a small Java™ utility that computes a dot product of two vectors; see Figure 6. The simulated performance of this hardware method on the JEM™ microprocessor is 70 times faster than the original software dot product method. We are currently working on automating the process. We expect the open system interface of our approach will enable the incorporation of other translation research for other software languages to VHDL.

Another final aspect of supportability is that software interfaces will never have to be changed due to changing FPGA technology. This characteristic is vital to avionics and defense products because of their long life times. Rockwell Collins often supports such products for twenty to thirty years.

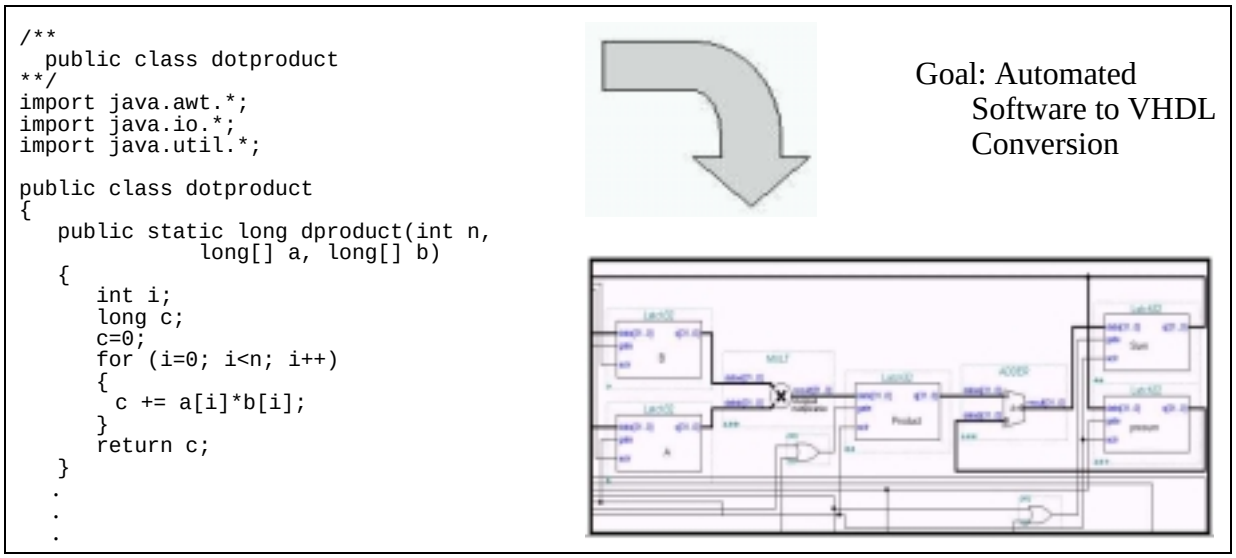


Figure 6. Hardware Method Production

7. Vision - Adaptable High Integrity Computing

Rockwell Collins is a supplier of high assurance systems for safety critical and secure applications in avionics and communications products. Our products demand increased performance and reduced power consumption and size while maintaining necessary system integrity. These demands place difficult constraints on the system designer. Advanced computing environments typically achieve high performance at the cost of reduced margin in both the electrical and the overall system complexity [DWJ98].

Reconfigurable computing is a technology that offers high performance computing while maintaining high system integrity. The traditional approaches to adaptable computing have been to employ reconfigurable logic elements to implement portions of the computing environment [LAJ98]. While these approaches may provide the desired performance enhancement, they do not address system integrity. The Rockwell Collins concept is to implement reconfigurable logic in an architectural environment that enforces protection and isolation of critical system resources and prevents any violation of system integrity.

Figure 7 illustrates our vision for the introduction of adaptive computing technologies in avionics, security, and defense products. Previous research and product development at Rockwell Collins has provided the experience, tools, and methodologies to support hardware/software co-design and design correctness. Our current activities focus on the research and development of the artifacts necessary for a secure reconfigurable architecture. Future research will use this secure architecture to create a high-integrity computing environment. We envision several product development efforts using this environment to produce unique, adaptive computing product for our military and commercial customers. Products such as flight control systems, classified applications, global positioning satellite, communication products, and entertainment can all benefit from this secure reconfigurable computing architecture.

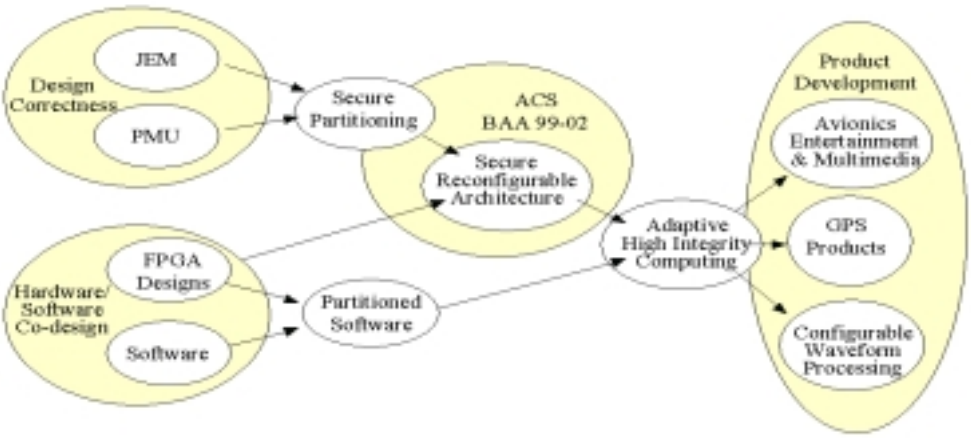


Figure 7. Adaptive High Integrity Computing Vision

8. Conclusion

Reconfigurable computing in the fashion advocated by DARPA and this paper is not used today in avionics and government systems. Microprocessor speeds typically limit the performance of many of today's systems. When additional performance is required, designers use separate ASICs or PLDs to accelerate specific functions. Secure and safety critical partitioning is done today with physical partitioning between multiple microprocessors. This limits exchange of data between processors to narrow bus bandwidths and effectively doubles the number of parts in a system. Using secure reconfigurable computing in these systems has the potential to decrease systems costs, reduce part counts, and increase reliability, maintainability, and serviceability. The major limitation of our approach over these existing approaches is their acceptance and validation by defense and security agencies. We hope to begin the process of collecting security requirements and address this limitation.

BIBLIOGRAPHY

[ACS98] Defense Advanced Project Agency, Information Technology Office, "BAA 99-02 Adaptive Computing System," Commerce Business Daily, October, 1998.

[AW97] Alexander Wolfe, "First Java-Specific MPU Rolls", Electronic Engineering Times, page 1, September 22, 1997.

[DAG98A] David A. Greve and Matthew M. Wilding, "The World's First Java Processor", January 1998. (Appears as "Stack-based Java a back-to-future step", Electronic Engineering Times, 12 January 1998)

[DAG98B] David A. Greve, "Symbolic Simulation of the JEM1 Microprocessor", Springer-Verlag Lecture Notes in Computer Science, Formal Methods in Computer-Aided Design -- FMCAD, 1998.

[DSH98] David S. Hardin, Matthew M. Wilding, and David A. Greve, "Transforming the Theorem Prover into a Digital Design Tool: From Concept Car to Off-Road Vehicle" Computer-Aided Verification – CAV '98, Springer-Verlag Lecture Notes in Computer Science volume 1427, 1998, available at http://www.pobox.com/users/hokie/docs/writings.html

[DWJ98] David W. Jensen, "Commercial and/or Proprietary Microprocessors in Rockwell Collins Products," a Rockwell Collins Vision Paper, September 1998.

[JMR98] John M. Rushby, “Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance,” Unpublished draft report, Computer Science Laboratory, SRI International, Menlo Park, CA, October 1998, available at http://www.csl.sri.com/~rushby/partitioning.html

[LAJ98] Luke A. Johnson and David W. Jensen, World Wide Web Search On Reconfigurable Computing, a Rockwell Collins White Paper, March 1998.

[MMW97] Matthew M. Wilding, "Robust Computer System Proofs in PVS", In LFM97: Fourth NASA Langley Formal Methods Workshop, C. Michael Holloway and Kelly J. Hayhurst, eds. NASA Conference Publication no. 3356, 1997. (Also http://atb-www.larc.nasa.gov/Lfm97)

[MMW98A] Matthew M. Wilding, David A Greve, and David S. Hardin, "Efficient Simulation of Formal Processor Models" unpublished manuscript, 1998, available at http://www.pobox.com/users/hokie/docs/writings.html

[MMW99] Matthew M. Wilding, David S. Hardin, and David A. Greve, “Invariant Performance: A Statement of Task Isolation Useful for Embedded Application Integration”, Dependable Computing for Critical Applications - DCCA-7, 1999, available at http://www.pobox.com/users/hokie/docs/writings.html

[OGO98] The Open Group, “What are open systems?” website brochure, November 24, 1998, available at http://www.opengroup.org/procurement/brochure/p1.htm,.

[RG98] Richard Goering, “C Level Design Tackles C-to-HDL”, EETimes Online, December 14, 1998. available at http://www.eet.com/story/OEG19981214S0042

[RSB96] Robert S. Boyer and Yuan Yu, "Automated Proofs of Object Code for a Widely Used Microprocessor", Journal of the ACM v. 43 n. 1, January 1996.

[SAS98] Scott A. Schoenig and David W. Jensen, "Conversion of Java Bytecodes to Hardware Schematics," a Rockwell Collins White Paper, October 30, 1998.

[SO98] S. Owre, N. Shankar, D.W.J. Stringer-Calvert, and J. Rushby, "The PVS System Guide", SRI Computer Science Lab technical report, September 98.

[SPM96] Steven P. Miller and Mandayam Srivas, "Formal Verification of the AAMP5 Microprocessor: A Case Study in the Industrial Use of Formal Methods", WIFT'95: Workshop on Industrial-Strength Formal Specification Techniques, April 1995.

[SPM99] Steven P. Miller, David A. Greve, Matthew M. Wilding, and Mandayam Srivas, "Formal Verification of the AAMP-FV Microcode", NASA Report NASA/CR-1999-208992, February 1999.

[WRB89A] William R. Bevier, "KIT: A Study in Operating System Verification", IEEE Transactions on Software Engineering v. 15 n. 11, November 1989.

[WRB89B] William R. Bevier, Warren A. Hunt Jr., J Strother Moore, and William D. Young, "An Approach to Systems Verification", Journal of Automated Reasoning v. 5 n. 4, December 1989.